PROTECTED CRYPTOGRAPHIC CALCULATION

[0001] The invention relates in general to the technical domain of cryptography and more especially to a procedure for improved protection of a cryptographic calculation against attacks. The invention is provided in particular for use in portable data carriers, which can be configured as smart cards in different forms of construction or as chip modules, for example.

[0002] The RSA method, described, e.g. in U.S. patent 4,405,829, is well known for the exchange of encrypted and/or signed data. According to the RSA method, a public key is used for encryption or signature verification and a secret private key for decryption or signature generation. Security of the RSA method is based on the fact that currently no efficient way of determining the prime factors p and q of a large number n with $n = p \cdot q$ is known. Whereas the so-called modulus n is published as part of the public key, the values p and q must be kept secret.

[0003] The calculation processes required for executing the RSA method are relatively complicated. For instance, the data to be processed have to be exponentiated with parameters of the private key during decryption or signature generation, for example. In particular for portable data carriers with their limited computing power, an implementation of the RSA method for decryption or signature generation is therefore frequently employed which uses the CRT (Chinese remainder theorem) and therefore is also designated as RSA-CRT method. By using the RSA-CRT method the required computing expenditure is reduced by approximately the factor 4.

[0004] The RSA-CRT method provides, instead of one complicated power calculation, to perform two far simpler exponentiations, the results of which are then combined into the decrypted data or the generated signature. Only the secret prime factor

p enters into the first of these calculations and only the secret prime factor q into the second calculation.

[0005] Attack scenarios have been proposed, in which exactly one of the two named RSA-CRT calculation branches is interfered with, e.g. by deliberate action of heat or radiation or by electrical pulses. If this succeeds, a multiple of the prime factor p, q, whose calculation branch has not been interfered with can be derived from the result of the overall calculation. In other words, conclusions can be drawn as to the private key by means of the described attack. This has potentially catastrophic consequences, because not only the decryption or signature generation just performed, but all the cryptographic operations executed using the private key are compromised.

[0006] The attack just mentioned is known by the name "fault attack" or "Bellcore attack" and described, e.g., in column 4 of U.S. patent 5,991,415. Likewise in U.S. patent 5,991,415 a method is disclosed, in which an additional factor j enters into the calculation to protect against this attack which takes place during the cryptographic calculation. However, as will be shown below, there are further possibilities of attack, against which nothing can be done by the method known from U.S. patent 5,991,415.

[0007] Said possibility of attack is particularly critical if the cryptographic calculation is executed by a processor of a portable data carrier, for example a smart card or a chip module. A first reason for this is that portable data carriers of this kind are often used for security-critical applications, e.g. in connection with financial transactions, access control or the signature of legally binding documents. Secondly, portable data carriers are typically in the possession of the attacker while the cryptographic calculation is being executed, so this person has every opportunity to influence the calculation and to spy on the results of the calculation.

[0008] The object of the invention is to provide a technique for particularly good protection of cryptographic calculations against attacks. In particular, attacks based on similar principles to the "Bellcore attack" described above should be prevented. In preferred configurations the protection according to the invention should advantageously cooperate with other protection methods.

[0009] According to the invention this object is completely or partly achieved by a method for protected execution of a cryptographic calculation with the features of claim 1, a method for determining a key for a cryptographic calculation with the features of claim 12 , a computer program product as claimed in claim 14 and a portable data carrier as claimed in claim 15. The dependent claims define preferred configurations of the invention. The order in which the method steps are listed in the claims should not be interpreted as a limitation of the range of protection; rather, configurations of the invention are provided, in which these method steps are executed completely or partly in a different order or completely or partly parallel or completely or partly interleaved.

[0010] The invention starts from the fundamental awareness that an attack similar to the above-described Bellcore attack is possible not only owing to interference with the calculation processes during the cryptographic calculation, but also by the cryptographic calculation being supplied with incorrect parameters. This can be done, for example, by transferring a false pointer address to the calculation routine, or by an external alteration of the contents of memory fields in which key parameters are contained. The inventors have realized that conclusions as to key parameters which need to be kept secret may possibly be drawn from the result of a cryptographic calculation supplied with parameters corrupted in this way.

[0011] According to the invention it is provided, to protect against an attack of this kind, to execute an integrity check of the key drawn on for the cryptographic calculation.

By this measure the attack can be identified and defended against, in that, e.g. the cryptographic calculation is terminated without issuing a result. The integrity check cannot normally rule out manipulation of the key parameters with absolute certainty; however, it should provide protection against said attack which is adequate for practical purposes. This implies that a simple range check with a fixed lower limit and a fixed upper limit would not be regarded as an integrity check in the sense of the present invention.

[0012] The integrity check is preferably configured in such a way that a manipulation in which a monitored key parameter is corrupted in a random way is identified with a probability bordering on certainty, e.g. with a probability greater than $1 - 10^{-3}$ or greater than $1 - 10^{-6}$ or greater than $1 - 10^{-9}$. While the integrity check in many configurations comprises only individual, particularly critical key parameters, it is preferably provided to monitor all the parameters of a key which needs to be kept secret. Different test methods can be executed in this case for individual parameters or groups of parameters in the course of the integrity check.

[0013] The methods used for the integrity check are in each case aimed at identifying corruption of the monitored key parameter or the monitored key parameters. In a preferred configuration, the integrity check effectively determines whether a key parameter is within a range of valid values, wherein the range is non-contiguous in that it has a plurality of gaps. This type of testing usually exists if the key parameter has been calculated during key generation from the value actually required for the cryptographic calculation and an additional, per se redundant safeguard value, as is the case, e.g. with checksum calculations.

[0014] While it can be provided that many or all of the key parameters are checked individually in each case, it is preferably determined in the integrity check whether at

least two key parameters are in a predetermined relationship to one another. The integrity check can include a multiplicative operation, which in the wording of the present document includes multiplication, division, exponentiation, a modulo calculation and a divisibility test.

[0015] It is preferably checked whether a key parameter or a value derived from it is evenly divisible by a safeguard value. In this case the key parameter is preferably extracted during key generation by multiplying the value actually required for the cryptographic calculation by the safeguard value. The safeguard value may be a component of the key or permanently preset.

[0016] The method according to the invention is suitable for all cryptographic calculations in which a cryptographic attack enables conclusions as to at least one second key parameter by corruption of at least one first key parameter. The invention is provided in particular for safeguarding the decryption or signature generation in an RSA method, preferably in an RSA-CRT method. In these cases the integrity check relates to the private RSA key. Corresponding possibilities of attack are expected to be found for further cryptographic calculations, which can then likewise be safeguarded in the manner according to the invention.

[0017] In preferred configurations, in the integrity check it is determined whether an exponent used in an exponentiation operation is evenly divisible by a safeguard value. These embodiments of the invention can be particularly advantageously combined with an exponent-masking method, as known from the international publication document WO 01/48974 A1. In further advantageous configurations the prime factors of the RSA method are multiplied by a masking parameter – alternatively or additionally to the exponent masking just mentioned – so that the result of the calculation can be checked for accuracy by means of an equality check modulo the masking parameter.

[0018] The computer program product according to the invention has program commands to implement the method according to the invention. A computer program product of this kind may be a physical medium, for example a semi-conductor memory or a diskette or a CD-ROM, on which a program for executing a method according to the invention is stored. The computer program product may, however, also be a non-physical medium, for example a signal communicated via a computer network. The computer program product may be provided in particular for use in connection with the production and/or initialization and/or personalization of smart cards or other data carriers.

[0019] In preferred configurations the computer program product and/or the portable data carrier are further developed with features corresponding to the above-described features and/or those mentioned in the dependent method claims.

[0020] Further features, advantages and objects of the invention emerge from the following detailed description of several embodiment examples and embodiment alternatives. Reference is made to the schematic drawings.

[0021] Fig. 1 shows an example of a flow diagram of a method for key calculation, illustrating a public and a private key.

[0022] Fig. 2 shows an example of a flow diagram of a cryptographic calculation method.

[0023] Fig. 3 shows an example of a flow diagram of a detail of the method of Fig. 2 in a modified configuration.

[0024] Fig. 4 shows an example of a flow diagram of a further embodiment example of the cryptographic calculation method.

[0025] The method illustrated in Fig. 1 serves for calculating a public key 10 and a private key 12, which are configured for use in an RSA method. The dotted arrows

indicate in each case which key parameter is generated by which method step. In connection with use of the pair of keys 10, 12 by portable data carriers (e.g. smart cards), the method can be executed in a secured environment, e.g. in the course of the initialization or personalization of the data carrier. The externally calculated pair of keys is then transmitted into the data carrier as part of the initialization or personalization data. It is alternatively possible for the method of Fig. 1 to be executed by the data carrier itself, in order to determine the pair of keys.

[0026] The public key 10 has as key parameters a modulus n and a public exponent e. The private key 12 is provided for RSA calculations using the Chinese remainder theorem, also designated here as RSA-CRT calculations. The private key 12 has as key parameters a first and a second prime factor p, q, a CRT coefficient pinv, a first and a second safeguard value sp, sq and also a safeguarded first and a safeguarded second CRT exponent dp, dq.

[0027] The method illustrated in Fig. 1 begins in a way known per se in step 14 with the random choice of two prime numbers with a length of, e.g. 1024 or 2048 bits each, which are stored in the private key 12 as the first and second prime factors p, q. In the following step 16 the modulus n of the public key 10 is calculated as the product of the two prime factors p, q. The public exponent e is determined in step 18 as a random number, which is relatively prime to the value $(p-1)\cdot(q-1)$. As in the present embodiment example the private key 12 is tailored to RSA-CRT calculations, in step 20 the modular inverse of p modulo q is calculated and entered into the private key 12 as the CRT coefficient pinv.

[0028] In step 22 the value d is calculated as the modular inverse of the public exponent e modulo $(p-1)\cdot(q-1)$. In RSA methods which do not use the Chinese remainder theorem d, as the private exponent, would be the main component of the private key. In

known RSA-CRT methods the two CRT exponents d mod (p-1) and d mod (q-1) would be used instead of d. In the present embodiment example the private key 12, on the other hand, contains values derived from said CRT exponents d mod (p-1) and d mod (q-1) by an additional safeguarding measure. This safeguarding measure is in this case, by way of example, the multiplication by one safeguard value in each case. Manipulation of the safeguarded values can be detected by a divisibility check. In embodiment alternatives other safeguarding measures are provided, e.g. a checksum formation or multiple passing of at least the important parameters to be passed.

[0029] Two random numbers with a length of, for example, 64 bits (8 bytes) each are generated in step 24 as safeguard values sp, sq. The safeguarded first CRT exponent dp is calculated in step 26 according to dp := (d mod (p-1)) · sp. Correspondingly, the safeguarded second CRT exponent dq is determined in step 28 by the calculation dq := (d mod (q-1)) · sq. Said values are all stored as parameters of the private key 12. This ends the determination of a private key 12 protected against manipulation.

[0030] In the present embodiment example the private key 12 substantially exists in the form of a data structure RSAPrivateCRTKey according to the conventions of the Java Card application programming interface. These conventions are described in the document "Java Card™ 2.1.1 Application Programming Interface", Revision 1.0, 18 May 2000, published by Sun Microsystems, Inc., USA, currently available at http://java.sun.com/ products/javacard/javacard21.html. The data structure provided there has fields DP1 and DQ1 for the unsafeguarded CRT exponents d mod (p-1) and d mod (q-1). In order to accommodate the private key 12 according to the present embodiment example in a data structure of this kind, it is provided in the present embodiment example that the values sp and dp are stored together in the field DP1 of the RSAPrivateCRTKey and that correspondingly the values sq and dq are stored together in the field DQ1. In Fig.

1 this is indicated by dotted lines. In embodiment variants the values sq and dq can also be stored in other fields of the RSAPrivateCRTKey or outside this data structure.

[0031] The embodiment examples described here differ slightly from the above-mentioned Java Card specification insofar as in the present case the modular inverse of p modulo q is contained in the private key 12 as CRT coefficient pinv. According to the Java Card specification, on the other hand, a CRT coefficient PQ is used which is the modular inverse of q modulo p. Modifications of the method described here are provided, in which the CRT coefficient PQ corresponding to the Java Card specification is a component of the private key 12. The ideas according to the invention can also be used for configurations of this kind without substantial alteration.

[0032] Fig. 2 shows a first configuration of a safeguarded RSA-CRT method which serves for decryption or signature generation. The method is provided to be executed by a processor of a portable data carrier, in particular a smart card or a chip module. The method is for this purpose implemented in the form of program commands for this processor, stored in a ROM or EEPROM of the data carrier. The private key 12 required for decryption or signature generation is likewise stored in the EEPROM of the data carrier.

[0033] When the method is invoked, a pointer to the private key 12 is transferred to the invoked decryption or signature generation routine. The inventors have recognized that a cryptographic attack can be executed in that individual parameters of the private key 12 are manipulated before the start of the decryption or signature generation. This can be done, e.g. by deliberate action on the EEPROM containing the private key 12 or by transferring an incorrect address to the RSA-CRT routine. An attack of this kind would have the extremely disadvantageous consequence that conclusions could be drawn from the result of the calculation – e.g. the decrypted data or the generated signature – as to the

values of the secret key parameters. This would compromise the pair of keys 10, 12 for all previous and future calculations.

[0034] In order to prevent a cryptographic attack of this kind, in the method of Fig. 2 an integrity check of the private key 12 is provided, which includes a series of several partial tests. In Fig. 2 the dotted arrows indicate which key parameters enter into the respective partial tests.

[0035] The method begins in step 30 with the partial test as to whether the CRT coefficient pinv contained in the private key 12 does actually represent the modular inverse to the first prime factor p modulo the second prime factor q. In other words it is checked whether the fixed relationship $p \cdot pinv = 1 \bmod q$ has been fulfilled. If this is not the case, an error jump takes place and the method is terminated. If the check is successful, it can be assumed that the key parameters p, q and pinv have not been manipulated and the method is continued.

[0036] In the calculation module 32, which now follows, a first auxiliary value y1 is determined as the result of the first of the two CRT calculation branches. The data x to be decrypted or signed, the first prime factor p and the first CRT exponent d mod p-1 enter into this calculation, wherein the latter value is not directly available, but has to be derived from the safeguarded first CRT exponent dp and the first safeguard value sp.

[0037] To check whether one of the two values sp, dp has been manipulated, firstly in step 34 a divisibility check is performed. If the safeguarded first CRT exponent dp is not evenly divisible by the first safeguard value sp, an error jump and termination of the method takes place again. If, on the other hand, the division comes out without remainder, it can be assumed with probability bordering on certainty that at least no random corruption of one of the two key parameters sp and dp has taken place. This divisibility check represents only a small additional computing expenditure, as the safeguard value sp

has only a relatively small bit length. Deliberate manipulation of the two parameters sp and dp with knowledge of the safeguarding mechanism provided could not, of course, be discovered by the method described here; though it is not currently imaginable how an attacker could bring about deliberate writing of this kind of new values into individual EEPROM cells of the data carrier.

[0038] If the integrity check with respect to parameters sp and dp in step 34 was successful, in step 36 the actual calculation of the first auxiliary value y1 is executed according to y1 := (x mod p)^(dp/sp). With respect to the exponent dp/sp it is of course normally possible to have recourse to the result of the division already calculated in step 34. As the safeguarding method in the present embodiment example simply consisted of multiplication by the first safeguard value sp – see step 26 in Fig. 1 – dp/sp = d mod (p-1) applies and thus y1 = (x mod p)^(d mod (p-1)). This is the desired result of the first CRT calculation branch.

[0039] A second calculation module 38 corresponds to the second CRT calculation branch. The method operates in the same way as in the first calculation module 32, wherein, however, the second prime factor q, the second safeguard value sq and the safeguarded second CRT exponent dq are drawn on. In step 40 the integrity check in respect of key parameters sq and dq follows again and in step 42 the second auxiliary value y2 is calculated according to the formula y2 := (x mod q)^(dq/sq).

[0040] In calculation step 44 concluding the method the overall result y, in other words the decrypted data or the calculated signature, is determined in a way known per se by combining the two CRT auxiliary values y1 and y2. The calculation performed here can be expressed as a formula as y := (((y2 - y1) · pinv) mod q) · p + y1. Different evaluation sequences can of course be chosen for the calculation steps performed by the processor of the data carrier. Dfferent variants of the RSA-CRT calculations of steps 36,

42 and 44 are generally known from the literature, which differ in particular in the way in which interim results are reduced to the respective modulo ranges. The idea according to the invention of the integrity check and the multiplicative safeguarding of the CRT exponents dp and dq proposed in the present embodiment example can be combined with all these variants.

[0041] The integrity check according to the present invention is directed in particular against a cryptographic attack which is executed chronologically before the RSA calculations – at the latest during passing of the parameters to the RSA routine. The check during passing of the parameters can also be done in a simple manner in that, as well as the parameters to be passed, redundant information related thereto, for example in the form of checksums, is also stored and, after the parameters have been passed, a stored checksum is compared with a checksum which has been newly calculated from the passed parameters. Alternatively, at least important parameters to be passed can be multiply passed and checked for identity after passing.

[0042] Further methods of attack are known, which aim at spying out the individual calculation steps in order to enable conclusions as to the key parameters which need to be kept secret. The exponent formation in steps 36 and 42 is particularly exposed to such attacks, because with normal implementations of the exponentiation operation the processor activity during the course of calculation depends considerably on the bit sequence of the exponent. This processor activity can be spied out by measuring the power consumption (SPA = simple power analysis or DPA = differential power analysis) or other signals, such as, e.g. electrical field strengths.

[0043] To protect against attacks of this kind, it was proposed in international patent publication WO 01/48974 A1 to divide the exponent with remainder by a random number and, instead of a single exponentiation operation, to perform three separate

exponentiations, wherein the whole-number quotient, the random number and the remainder determined during the division are used as exponents. This method is described in detail in said patent publication, the contents of which are hereby fully incorporated into the present document.

[0044] It is a particular advantage of the safeguarding method according to the present invention that it can easily be combined with the masking method designated as "exponent blinding" according to WO 01/48974 A1, wherein the division required in any case for the masking method can also be used for the safeguarding method according to the present invention. The method according to the invention can in this way be implemented with very little extra expenditure.

[0045] Fig. 3 shows the method steps of a calculation module 32', which is modified in respect of calculation module 32 of Fig. 2 in such a way that it additionally applies the technique of exponent blinding known per se from WO 01/48974 A1. For this, firstly in step 46 a random number r with a length of, for example, 64 bits (8 bytes) is chosen. In step 48 a division with remainder is performed, to divide the safeguarded CRT exponent dp into the factors dp1 and r·sp and the remainder dp2; dp = dp1·r·sp + dp2 applies. In contrast to the method known from WO 01/48974 A1, in step 48 the value r·sp is therefore used as divisor, and not the random number r.

[0046] In steps 50 and 52 the first two exponentiation operations are now performed, in that the basic value x mod p is first exponentiated by the random number r and the thus obtained interim result y11 is then exponentiated by the whole-number quotient dp1. For the result y12, y12 = ((x mod p)^r)^dp1 = (x mod p)^(r·dp1) thus applies. The safeguard value sp has not entered into steps 50 and 52, as to this extent the multiplication serving to safeguard the CRT exponent dp has already been reversed in connection with the division 48.

[0047] In step 54 a divisibility check now takes place – analogously to step 34 in Fig. 2 – to ensure the integrity of the key parameters sp and dp. In contrast to step 34 in Fig. 2, here, however, it is not the safeguarded CRT exponent dp that is divided by sp, but the division remainder dp2. As dp2 differs from dp only by a multiple of r·sp – and thus by a multiple of sp – the two checks are of equal value. The calculation expenditure arising from execution of step 54 is, however, considerably less than in the calculation of step 34 in Fig. 2, owing to the far shorter dividend dp2. Moreover, the whole-number division result dp2/sp is required in the following step 56. The division and divisibility check in step 54 represents in comparison with the known method according to WO 01/48974 A1 the only additional computing expenditure.

[0048] If dp2 is not an even multiple of sp, the method is terminated in step 54 by an error jump. Otherwise, in step 56 a further interim value y13 is calculated according to y13 := (x mod p)^(dp2/sp). The product y12·y13 is determined in step 58 as the result y1 of calculation module 32'. This result is identical to the first auxiliary value y1 according to step 36 of Fig. 2, because:

$$
\begin{aligned}
y1 &= y12 \cdot y13 \\
&= ((x \bmod p)^{(r \cdot dp1)}) \cdot ((x \bmod p)^{(dp2/sp)}) \\
&= (x \bmod p)^{((r \cdot dp1)+(dp2/sp))} \\
&= (x \bmod p)^{(dp/sp)}
\end{aligned}
$$

applies.

[0049] The entire RSA-CRT method in the particularly protected embodiment variant described here starts with an integrity check of parameters p, q and pinv by step 30 shown in Fig. 2. This is followed by the steps of calculation module 32' according to Fig. 3 as first CRT calculation branch, in order to determine the first auxiliary value y1. The method shown in Fig. 3 is likewise used to calculate the second auxiliary value y2, wherein, of course, the key parameters p, sp and dp are replaced by q, sq and dq. The

random number r can either be taken over from the first course of the calculation module 32' or newly determined. The final result y is finally calculated by combining the two auxiliary values y1 and y2, as in step 44 of Fig. 2.

[0050] The method shown in Fig. 4 provides an additional checking step, in which a further masking parameter j is drawn on. A first calculation block 60 corresponds approximately to step 30 in Fig. 2. In step 62 the masking parameter j is chosen as a random prime number with a length of, for example, 32 bits (4 bytes). The prime factors p and q are multiplied by the masking parameter j in steps 64 and 66 to obtain masked prime factors p' or q'. In step 68 a test takes place to check the integrity of key parameters p, q and pinv. If p'·pinv = j mod q' applies, the method is continued; otherwise an error jump takes place.

[0051] In a second calculation block 70 a first auxiliary value y1 is determined according to the formula y1 := (x^(dp/sp)) mod p'. The first auxiliary value y1 substantially corresponds to the first auxiliary value of the embodiment examples of Fig. 2 and Fig. 3, wherein, however, p' is drawn on instead of p for the modulo calculation. In detail the calculation takes place in different embodiment variants either as in calculation module 32 of Fig. 2 or as in calculation module 32' of Fig. 3. In both cases a divisibility check is performed to ensure the integrity of key parameters sp and dp.

[0052] A third calculation block 72 corresponds to the second calculation block 70 with the difference that instead of dp, sp and p', the values dq, sq and q' are drawn on to calculate a second auxiliary value y2. Again the third calculation block 72 can be configured either like calculation module 38 in Fig. 2 or analogously to the representation in Fig. 3. The integrity of key parameters sq and dq is checked by a divisibility test in the third calculation block 72.

[0053] Step 74 relates to the calculation of an interim result y' according to the formula y' := [((y2 - y1) · pinv) mod q'] · p + y1. This corresponds approximately to step 44 in Fig. 2. In step 76 a further test takes place, which relates the calculations so far to one another and identifies interfered with calculation sequences. It is checked whether the following equality relationship modulo j applies:

$$y' \bmod j = [\ ((x^{(dq/sq)}) \bmod j - (x^{(dp/sp)}) \bmod j) \cdot pinv \cdot p + (x^{(dp/sp)}) \bmod j\ ] \bmod j$$

If this equation is not fulfilled, an error termination takes place. Otherwise, the method is concluded in step 78 with the calculation of the final result y according to y := y' mod n, wherein n is the modulus with n = p·q. Even greater improved protection against cryptographic attacks is achieved in the method according to Fig. 4 by further checking of the calculation sequence in step 76.